**Agenda**

- ✓ **Waits in Selenium**
  - ■ **ImplicitWait**
  - ■ **ExplicitWait**
  - ■ **PageLoadTimeOut**
- ✓ **Drawbacks of Thread.sleep()**
- ✓ **Difference between findElement and findElements( )**

*****Waits in Selenium**
**Implicit wait**

If we don't apply implicit wait, the findElement() and findElements() will try to locate the web elements, if the web element is not located, then immediately NoSuchElementException would be thrown.
To Inform the findElement( )/findElements( ) to wait for certain amount of time before it throws NoSuchElementException we should use implicitWait

**Syntax**
**driver.manage().timeOuts().implicitlyWait(Duration.ofSeconds(number_of_seconds));**

Meaning of implicit wait

Its not that we should wait for 10 seconds before performing the actions, It's the max time the findElement() / findElements( )should wait before it throws NoSuchElementException

If the web element is located even before 10 seconds, then the action would be performed as soon as the web element is found. The max time it would wait for web element to load is 10 sec.

## Ex:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import io.github.bonigarcia.wdm.WebDriverManager;

public class ImplicitWaitExample {

	public static void main(String[] args) {
		try {
			WebDriverManager.chromedriver().setup();
			ChromeDriver driver = new ChromeDriver();
			Thread.sleep(5000);
			driver.manage().window().maximize();
			Thread.sleep(2000);
			driver.manage().timeouts().implicitlyWait(Duration.
			ofSeconds(10));
			driver.get("https://opensource-
			demo.orangehrmlive.com/web/index.php/auth/login");
			WebElement userName =
			driver.findElement(By.name("username"));
			userName.sendKeys("Admin");
			Thread.sleep(3000);
			driver.quit();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

## Explicit wait

In case of Implicit wait the wait is applicable for all the web elements on the page. But in case of explicit wait, we can apply waits on individual web elements.

## Syntax

In order to add explicit wait follow the below steps

1) Create an object of a class WebDriverWait( )
2) Pass the ChromeDriver reference and the time duration as the parameters to WebDriverWait()

Ex:

```
import java.time.Duration;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import io.github.bonigarcia.wdm.WebDriverManager;

public class ExplicitWaitExample {

	public static void main(String[] args) {
	try {
	WebDriverManager.chromedriver().setup();
	ChromeDriver driver = new ChromeDriver();
	Thread.sleep(4000);
	driver.manage().window().maximize();
	driver.get("https://opensource-
	demo.orangehrmlive.com/web/index.php/auth/login");
	WebDriverWait wait = new WebDriverWait(driver,
	Duration.ofSeconds(10));
```

```
        WebElement userName =
        wait.until(ExpectedConditions.visibilityOfElementLocated(By.nam
        e("username")));
        userName.sendKeys("Admin");
        } catch (Exception e) {
              System.out.println(e.getMessage());
        }
        }
}
```

In the above example, we are specifically applying the wait on the username field. We are telling selenium that it should wait for max of 10 seconds to perform the actions on the username field. If the web element is located within 10 seconds, immediately perform the action else, throw the exception

**Difference between implicitWait and explicitWait**

| **Implicit Wait** | **Explicit Wait** |
|---|---|
| Its applied on all the web elements | Its applied on a specific web element |
| Syntax:<br>driver.manage()<br>.timeouts().implicitlyWait(Duration.ofSeconds(10)) | WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));<br>WebElement userName = wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("username")));<br>; |
| Its very simple to use. | It is applied on individual |

| | |
|---|---|
| If we add it once, it will be applicable for all web elements | elements, the syntax looks complex |
| In case of implicitWait() **NoSuchElementException** is thrown when the web element is not located | In case of explicit wait, **TimoutException** would be thrown when the web element is not located |

**PageLoadTimeOut**

Its applying the wait for the entire page to get loaded. If within the specified time if the page gets loaded then no issues, but if the page doesn't get loaded with in the specified time, the PageLoadTimeOut would stop the webpage from loading.

Its mainly used to check the performance of the webpage.

**Ex**

```
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import io.github.bonigarcia.wdm.WebDriverManager;

public class PageLoadTimeOutExample {
	public static void main(String[] args) {
		try {
			WebDriverManager.chromedriver().setup();
			ChromeDriver driver = new ChromeDriver();
			Thread.sleep(5000);
			driver.manage().window().maximize();
			driver.manage().timeouts().pageLoadTimeout(Duration.
			ofSeconds(2));
			driver.get("https://opensource-
			demo.orangehrmlive.com/web/index.php/auth/login");
			WebElement userName =
			driver.findElement(By.name("username"));
			userName.sendKeys("Admin");
		} catch (Exception e) {
			System.out.println(e);
		}
	}
}
```

## Drawbacks of Thread.sleep( )

If we use Thread.sleep( ), It will forcefully blocks the execution of the program for the specified amount of time, irrespective of whether the web element is located or not, the program execution would be blocked until the time duration is completed. Thread.sleep() would reduce the efficiency of the program.

## findElements( )

In order to find multiple elements on the webpage, we can use findElements( ).

Ex:

## Scenario to be automated

1) Open any browser
2) Navigate to www.amazon.in
3) Identify all the links on the amazon.in website and display the link texts.

```
import java.time.Duration;
import java.util.Iterator;
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import io.github.bonigarcia.wdm.WebDriverManager;

public class FindElementsExample {

	public static void main(String[] args) {
		try {
			WebDriverManager.chromedriver().setup();
			ChromeDriver driver = new ChromeDriver();
			driver.manage().timeouts().implicitlyWait(Duration.
			ofSeconds(10));
			Thread.sleep(3000);
			driver.manage().window().maximize();
			driver.get("https://www.amazon.in");
			List<WebElement> links =
			driver.findElements(By.tagName("a"));
			Iterator<WebElement> itr = links.iterator();
			while(itr.hasNext()) {
			System.out.println(itr.next().getText());
			}
		} catch (Exception e) {
			System.out.println(e);
		}
	}
}
```

## O/P

## All link texts will be displayed

## Difference between findElement vs findElements()

| **findElement** | **findElements** |
|---|---|
| It can find maximum of 1 web element | It can find any number of web elements |
| The return type of findElement is WebElement | The return type of findElements List<WebElement> |
| If in case the web element is not found then NoSuchElementException would be thrown | If in case the web element is not found then it doesn't throw any exception, it return an empty list(Array) |
| Syntax:<br>WebElement el = driver.findElement(…); | Syntax:<br>List<WebElement> list = driver.findElements(…); |